Source Code: kirby_lab03.c

```
/*****************************************************************************/
/*                                                                           */
/*      ECE 344L    -    Microprocessors    -    Spring 2020                 */
/*                                                                           */
/*      kirby_lab03.c  -  Digital I/O and Finite State Machine               */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/*      Author: David Kirby                                                  */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/*      File Description:                                                    */
/*          Implements a finite state machine using the LEDs and buttons on  */
/*          the chipKIT MX7 board.                                           */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/*      Revision History:                                                    */
/*          Original Source Code by: E.J. Nava, 9/23/18                      */
/*          Modified Code by: David Kirby, 01-Mar-2020                       */
/*                                                                           */
/*****************************************************************************/

#include <plib.h>

/* ------------------------------------------------------------------------- */
/*                            Configuration Bits                             */
/* ------------------------------------------------------------------------- */

// Configure MX7 board for debugging
#pragma config ICESEL = ICS_PGx1

// SYSCLK = 80 MHz (8 MHz Crystal/ FPLLIDIV * FPLLMUL / FPLLODIV)
// Primary Osc w/PLL (XT+,HS+,EC+PLL)

#pragma config FPLLMUL = MUL_20, FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1
#pragma config POSCMOD = EC, FNOSC = PRIPLL, FPBDIV = DIV_8
#pragma config FSOSCEN = OFF            // Secondary oscillator enable
#define SYS_FREQ (80000000L)

// *** these are preconfigured on the MX4 Board for a clock frequency of 80MHz
// *** and a PBCLK value of 10MHz.

/* ------------------------------------------------------------------------- */
/*                           Forward Declarations                            */
/* ------------------------------------------------------------------------- */

void DeviceInit();
void DelayInit();
void DelayMs(int cms);
```

```
53  void DisplayInit(int coins);
54
55
56  /* ---------------------------------------------------------------------- */
57  /*                                Definitions                             */
58  /* ---------------------------------------------------------------------- */
59
60  #define cntMsDelay  10000           //timer 1 delay for 1ms
61
62  /* ---------------------------------------------------------------------- */
63  /*                                   Main                                 */
64  /* ---------------------------------------------------------------------- */
65
66  int main()
67  {
68          int button_in12 = 0;
69          int button_in3 = 0;
70          int coins = 0;
71          int msdelay = 100;
72
73          //Set LD1 through LD4 as digital output
74          DeviceInit();
75          //Initialize timer for delay
76          DelayInit();
77
78          /* Perform the main application loop*/
79          while (1)
80          {
81                  // Read buttons
82                  button_in12 = PORTReadBits (IOPORT_G, BIT_6|BIT_7);
83                  button_in3  = PORTReadBits (IOPORT_A, BIT_0);
84
85                  if (button_in12 != 0)
86                  {
87                          // drive both LD1 and LD2 high if both buttons pressed
88                          if (((button_in12 & 0x0040) != 0) &&
89                              ((button_in12 & 0x0080) != 0))
90                                  coins = coins+15;
91                          else
92                          {
93                                  //drive LD1 high if only BTN1 pressed
94                                  if ((button_in12 & 0x0040) !=0) // BTN1 pressed?
95                                          coins = coins+5;
96                                  //drive LD2 high if only BTN2 pressed
97                                  if ((button_in12 & 0x0080) != 0) // BTN2 pressed
98                                          coins = coins+10;
99                          }
100                 }
101                 // Handle BTN3 separately
102                 if(button_in3 !=0)
103                 {
104                     coins=0;
105                     PORTWrite(IOPORT_G,BIT_12|BIT_13|BIT_14);
106                     DelayMs(msdelay);
```

```
                    PORTClearBits(IOPORT_G,BIT_12|BIT_13| BIT_14|BIT_15);
                }
                //Initialize display
                DisplayInit(coins);

        }

}

/* ------------------------------------------------------------------------ */
/*    DisplayInit()
**
**    Parameters:
**        coins             -amount of money entered
**        delay             -delay between blinks
**
**    Return Value:
**        none
**
**    Errors:
**        none
**
**    Description:
**        Set display state based on amount of money entered
/* ------------------------------------------------------------------------ */

void DisplayInit(int coins)
{
        int msdelay = 230;
        int timeout=0;

        switch (coins)
        {
        case  5:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_12);   //001
                break;
        case 10:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_13);   //010
                break;
        case 15:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_12|BIT_13);   //011
                break;
        case 20:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
```

```
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_14);                                   //100
                break;
        case 25:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_12|BIT_14);                            //101
                break;
        case 30:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                while(timeout<3)
                {
                      DelayMs(msdelay);
                      PORTWrite (IOPORT_G, BIT_15);                             //111
                      DelayMs(msdelay);
                      PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                      timeout++;
                }
                main();
                break;
        case 35:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                while(timeout<3)
                {
                        DelayMs(msdelay);
                        PORTWrite (IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);//111+
                        DelayMs(msdelay);
                        PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                        timeout++;
                }
                main();
                break;
        default:
                PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                //Debug LEDs - send them all high at first, then trigger
                //PORTWrite (IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
        }
}

/* ------------------------------------------------------------------------ */
/*    DeviceInit()
**
**    Parameters:
**        none
**
**    Return Value:
**        none
**
**    Errors:
**        none
**
```

```
215 **    Description:
216 **        Set LD1 through LD4 as digital output
217 /* ------------------------------------------------------------------------ */
218
219 void DeviceInit()
220 {
221         // On MX7 board, disable JTAG function
222         DDPCONbits.JTAGEN = 0;
223
224         //On MX7 LED1 is on RG12
225         //       LED2 is on RG13
226         //       LED3 is on RG14
227         //       LED4 is on RG15
228         //Set ports for onboard LEDs to outputs & clear them
229         PORTSetPinsDigitalOut (IOPORT_G, BIT_12|BIT_13| BIT_14|BIT_15);
230         PORTClearBits(IOPORT_G, BIT_12|BIT_13| BIT_14|BIT_15);
231         //Set ports for onboard BTNs as inputs
232         PORTSetPinsDigitalIn (IOPORT_G, BIT_6 | BIT_7);
233         PORTSetPinsDigitalIn (IOPORT_A, BIT_0);
234 }
235
236 /* ------------------------------------------------------------------------ */
237 /*    DelayInit
238 **
239 **    Parameters:
240 **        none
241 **
242 **    Return Value:
243 **        none
244 **
245 **    Errors:
246 **        none
247 **
248 **    Description:
249 **        Initialized the hardware for use by delay functions. This
250 **        initializes Timer 1 to count at 10Mhz.
251 /* ------------------------------------------------------------------------ */
252
253 void DelayInit()
254 {
255         unsigned int tcfg;
256
257         /* Configure Timer 1 to count a 10MHz with a period of 0xFFFF*/
258         tcfg = T1_ON|T1_IDLE_CON|T1_SOURCE_INT|T1_PS_1_1|T1_GATE_OFF|T1_SYNC_EXT_OFF;
259         OpenTimer1(tcfg, 0xFFFF);
260 }
261
262 /* ------------------------------------------------------------------------ */
263 /*    DelayMs
264 **
265 **    Parameters:
266 **        cms             - number of milliseconds to delay
267 **
268 **    Return Value:
```

```
269 **         none
270 **
271 **    Errors:
272 **         none
273 **
274 **    Description:
275 **         Delay the requested number of milliseconds. Uses Timer1.
276 /* ------------------------------------------------------------------------ */
277
278 void DelayMs(int cms)
279 {
280         int ims;
281
282         for (ims=0; ims<cms; ims++)
283         {
284                 WriteTimer1(0);     // reset timer
285                 while (ReadTimer1() < cntMsDelay); // wait for interval of 1 mS
286         }
287
288 }
```